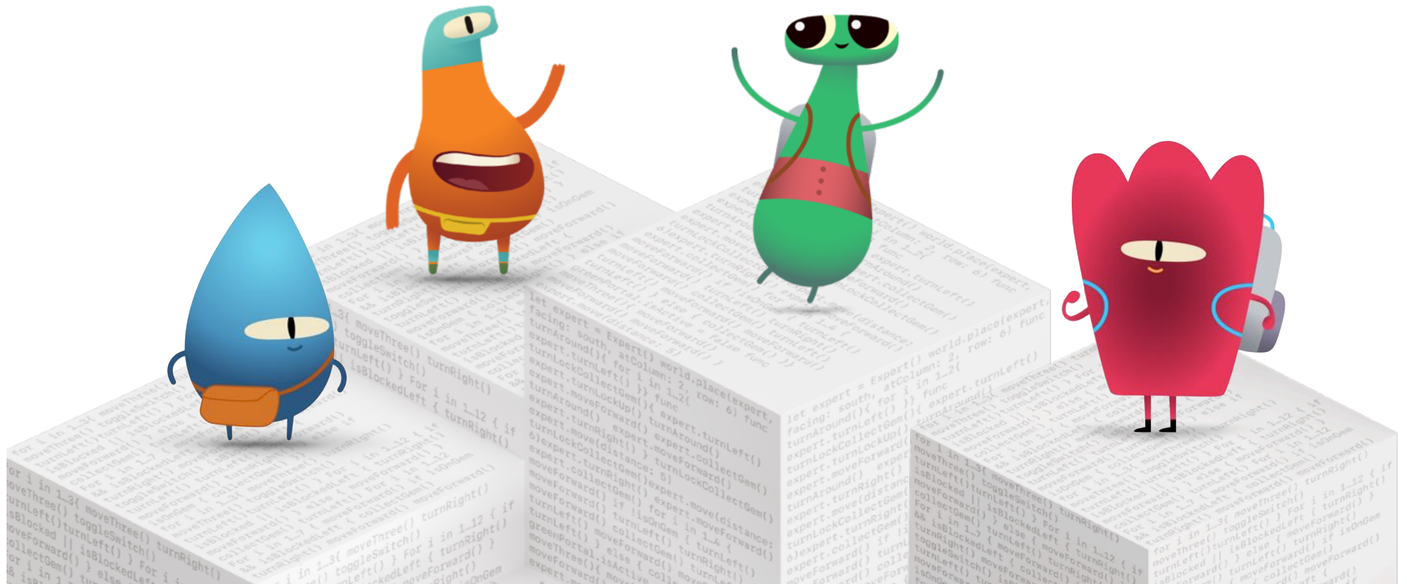
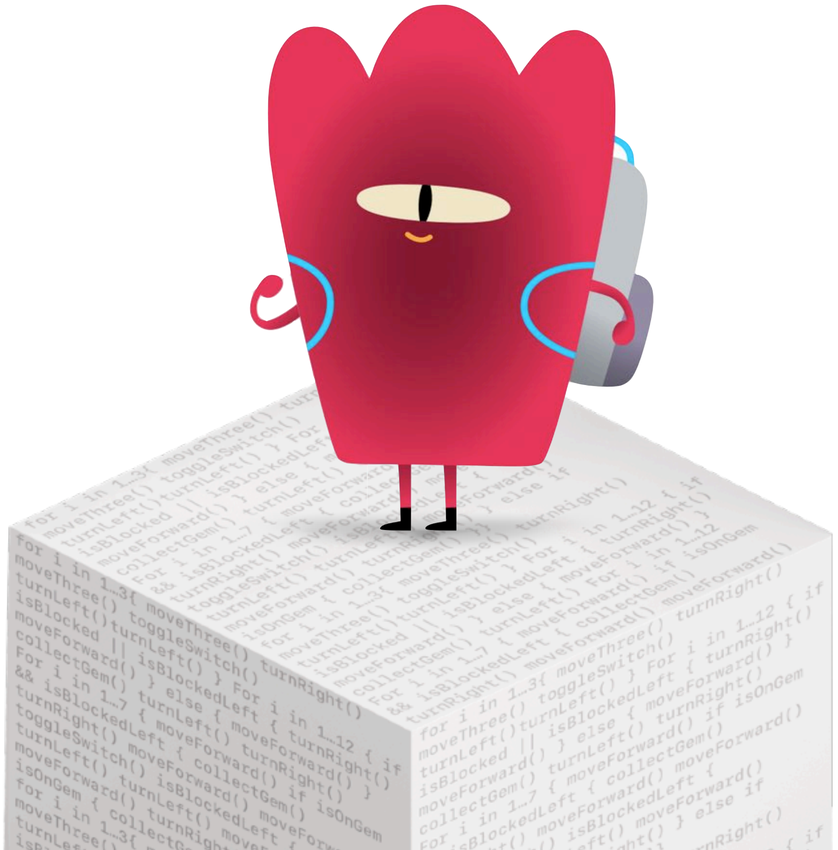
Swift Coding Club

Special Session



Session 1

Intro to Swift Student Challenge + Swift Fundamentals



Introduction to Swift

Intro to Swift Student Challenge + Swift Fundamentals

What is Swift?

Introduction to Swift

Intro to Swift Student Challenge + Swift Fundamentals

Let’s create a playground!

Create a Playground in Swift Playgrounds

Introduction to Swift

Intro to Swift Student Challenge + Swift Fundamentals

Create a Playground in Xcode

Constants and Variables

Constants and variables

Associate a name with a value

Defining a constant or variable

- Allocates storage for the value in memory
- Associate the constant name with the assigned value

Constants

Defined using the `let` keyword

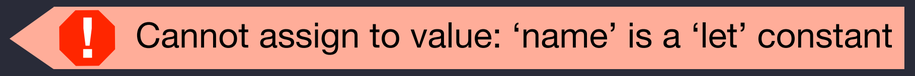
```
let name = "John"
```

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
```

```
let defaultScore = 100
var playerOneScore = defaultScore
var playerTwoScore = defaultScore

print(playerOneScore)
print(playerTwoScore)

playerOneScore = 200
print(playerOneScore)
```

```
100
100
200
```

Naming constants and variables

Rules

No mathematical symbols

No spaces

Can’t begin with a number

```
let π = 3.14159
let 一百 = 100
let 🎲 = 6
let mañana = "Tomorrow"
let anzahlDerBücher = 15 //numberOfBooks
```

Naming constants and variables

Best practices

1. Be clear and descriptive

   ✗ `n` ✓ `firstName`

2. Use camel case when multiple words in a name

   ✗ `firstname` ✓ `firstName`

Comments

```
// Setting pi to a rough estimate
let π = 3.14
```

```
/* The digits of pi are infinite,
so instead I chose a close approximation.*/
let π = 3.14
```

Types

```
struct Person {
  let firstName: String
  let lastName: String

  func sayHello() {
    print("Hello there! My name is \(firstName) \(lastName).")
  }
}
```

```
struct Person {
  let firstName: String
  let lastName: String

  func sayHello() {
    print("Hello there! My name is \(firstName) \(lastName).")
  }
}

let aPerson = Person(firstName: "Jacob", lastName: "Edwards")
let anotherPerson = Person(firstName: "Candace", lastName: "Salinas")

aPerson.sayHello()
anotherPerson.sayHello()

Hello there! My name is Jacob Edwards.
Hello there! My name is Candace Salinas.
```

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
```

Cannot assign value of type ‘String’ to type ‘Int’

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

Cannot assign value of type ‘Double’ to type ‘Int’

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
}
```

Required values

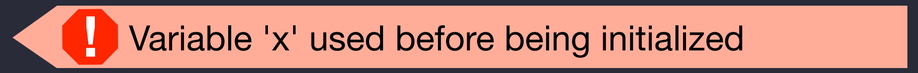
```
var x
```

Required values

```
var x: Int
```

Required values

```
var x: Int
print(x)
```



Required values

```
var x: Int
x = 10
print(x)
```

```
10
```

Numeric literal formatting

```
var largeUglyNumber = 1000000000
var largePrettyNumber = 1_000_000_000
```

Operators

Assign a value

Use the = operator to assign a value

```
var favoritePerson = "Luke"
```

Use the = operator to modify or reassign a value

```
var shoeSize = 8
shoeSize = 9
```

Basic arithmetic

You can use the `+`, `−`, `∗`, and `/` operators to perform basic math functions

```
var opponentScore = 3 * 8
var myScore = 100 / 4
```

You can also use the value of other variables

```
var totalScore = opponentScore + myScore
```

Or you can use the current variable you’re updating

```
myScore = myScore + 3
```

Basic arithmetic

Use Double values for decimal point precision

```
let totalDistance = 3.9
var distanceTravelled = 1.2
var remainingDistance = totalDistance - distanceTravelled
print(remainingDistance)
```

```
2.7
```

Basic arithmetic

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

```
12
```

Compound assignment

```
var myScore = 10
myScore = myScore + 3
```

```
myScore += 3
myScore -= 5
myScore *= 2
myScore /= 2
```

Order of operations

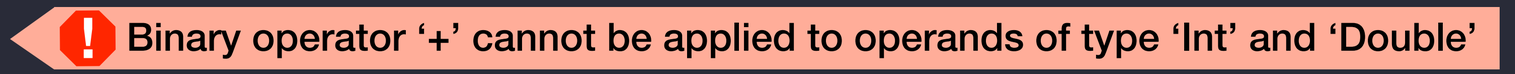
1. `(` `)`
2. `*` `/`
3. `+` `-`

```
var x = 2
var y = 3
var z = 5
print(x + y * z)
print((x + y) * z)
```

```
17
25
```

Numeric type conversion

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

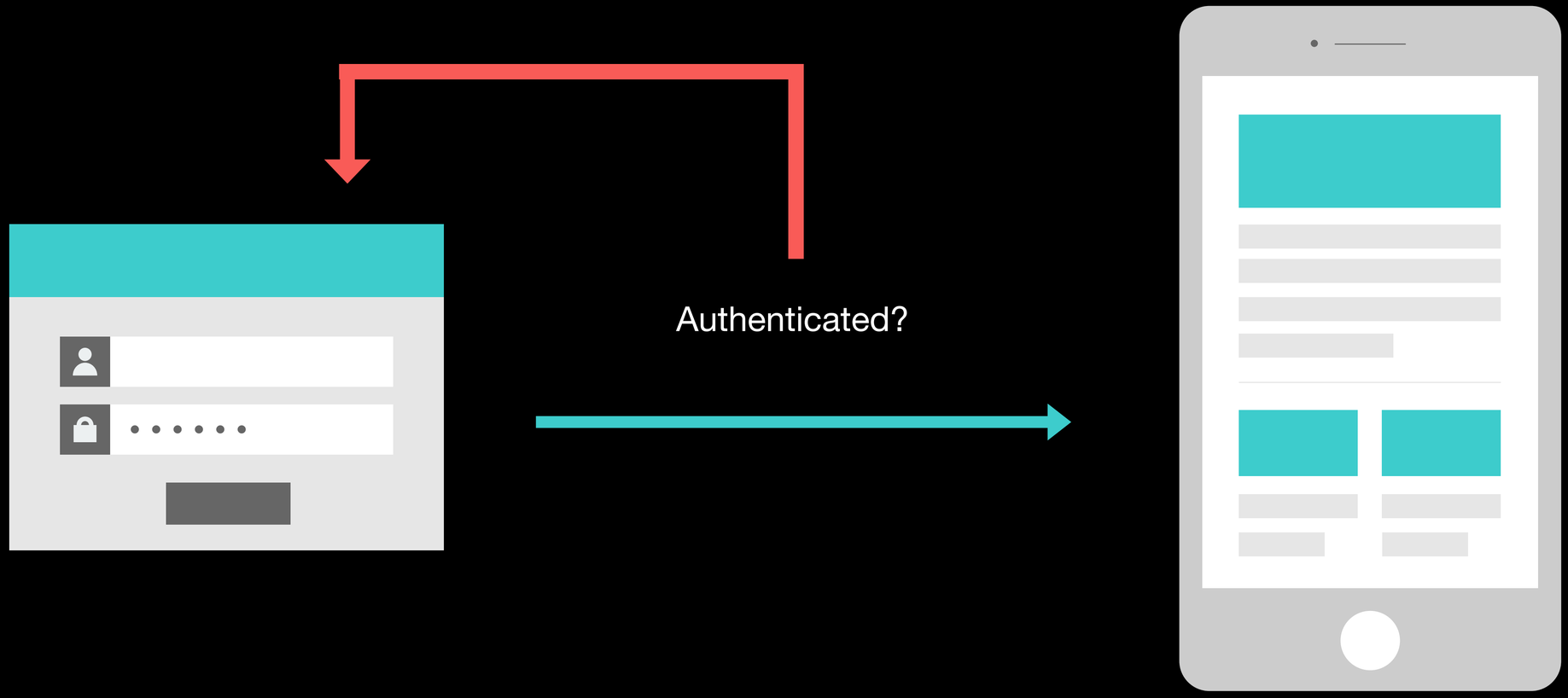


Numeric type conversion

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

Control Flow

Conditional flow



Logical operators

Operator	Description
`==`	Two items must be equal
`!=`	The values must not be equal to each other
`>`	Value on the left must be greater than the value on the right
`>=`	Value on the left must be greater than or equal to the value on the right
`<`	Value on the left must be less than the value on the right
`<=`	Value on the left must be less than or equal to the value on the right
`&&`	AND—The conditional statement on the left and right must be true
`||`	OR—The conditional statement on the left or right must be true
`!`	Returns the opposite of the conditional statement immediately following the operator

if statements

```
if condition {
    code
}
```

```
let temperature = 100
if temperature >= 100 {
  print("The water is boiling.")
}
```

```
The water is boiling
```

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

```
let number = 1000
let isSmallNumber = number < 10
```

```
let speedLimit = 65
let currentSpeed = 72
let isSpeeding = currentSpeed > speedLimit
```

Boolean values

NOT

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

```
It is not snowing.
```

Boolean values

AND

```
let temperature = 70
if temperature >= 65 && temperature <= 75 {
  print("The temperature is just right.")
} else if temperature < 65 {
  print("It's too cold.")
} else {
  print("It's too hot.")
}
```

```
The temperature is just right.
```

Boolean values

OR

```
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 0:
    print(“Missing something?”)
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

Multiple conditions

```
let character = "z"

switch character {
case "a", "e", "i", "o", "u" :
    print("This character is a vowel.")
default:
    print("This character is not a vowel.")
}
```

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?")
}
```

switch challenge

Rewrite the following using a switch statement:

```
let temperature = 70
if temperature >= 65 && temperature <= 75 {
  print("The temperature is just right.")
} else if temperature < 65 {
  print("It's too cold.")
} else {
  print("It's too hot.")
}
```

Hint: The smallest possible value for an integer is `Int.min`

switch challenge

Solution

```
let temperature = 76
switch temperature {
case Int.min...64:
    print("It's too cold.")
case 65...75:
    print("The temperature is just right.")
default:
    print("It's too hot.")
}
```

Ternary operator

```
var largest: Int
let a = 15
let b = 4

if a > b {
  largest = a
} else {
  largest = b
}
```

Ternary operator

?:

```
variable = condition ? true_value : false_value
```

```
var largest: Int
let a = 15
let b = 4

largest = a > b ? a : b
```

Functions

Functions

```
tieMyShoes()
```

```
makeBreakfast(food: "scrambled eggs", drink: "orange juice")
```

Functions

Defining a function

```
func functionName (parameters) -> ReturnType {
    // Body of the function
}
```

```
func displayPi() {
  print("3.1415926535")
}

displayPi()
```

```
3.1415926535
```

Parameters

```
func triple(value: Int) {
  let result = value * 3
  print("If you multiply \(value) by 3, you'll get \(result).")
}

triple(value: 10)
```

```
If you multiply 10 by 3, you'll get 30.
```

Parameters

Multiple parameters

```
func multiply(firstNumber: Int, secondNumber: Int) {
  let result = firstNumber * secondNumber
  print("The result is \(result).")
}

multiply(firstNumber: 10, secondNumber: 5)
```

```
The result is 50.
```

Return values

```
func multiply(firstNumber: Int, secondNumber: Int) -> Int {
  let result = firstNumber * secondNumber
  return result
}
```

Return values

```
func multiply(firstNumber: Int, secondNumber: Int) -> Int {
 return firstNumber * secondNumber
}
```

```
let myResult = multiply(firstNumber: 10, secondNumber: 5)
print("10 * 5 is \(myResult)")
```

```
print("10 * 5 is \(multiply(firstNumber: 10, secondNumber: 5))")
```

```
func multiply(firstNumber: Int, secondNumber: Int) -> Int {
 firstNumber * secondNumber
}
```

Argument labels

```
func sayHello(firstName: String) {
  print("Hello, \(firstName)!")
}

sayHello(firstName: "Amy")
```

Argument labels

```
func sayHello(to: String, and: String) {
  print("Hello \(to) and \(and)")
}

sayHello(to: "Luke", and: "Dave")
```

Argument labels

External names

```
func sayHello(to person: String, and anotherPerson: String) {
  print("Hello \(person) and \(anotherPerson)")
}

sayHello(to: "Luke", and: "Dave")
```

Argument labels

Omitting labels

```
print("Hello, world!")
```

```
func add(_ firstNumber: Int, to secondNumber: Int) -> Int {
    firstNumber + secondNumber
}

let total = add(14, to: 6)
```

Default parameter values

```
func display(teamName: String, score: Int = 0) {
  print("\(teamName): \(score)")
}

display(teamName: "Wombats", score: 100)
display(teamName: "Wombats")
```

```
Wombats: 100
Wombats: 0
```

Loops

Loops

`for`
`while`

for loops

```
for index in 1...5 {
  print("This is number \(index)")
}
```

for loops

```
for _ in 1...5 {
  print("Hello!")
}
```

for loops

```
let names = ["Joseph", "Cathy", "Winston"]
for name in names {
  print("Hello \(name)")
}
```

```
for letter in "ABCDEFG" {
  print("The letter is \(letter)")
}
```

for loops

```
for (index, letter) in "ABCDEFG".enumerated() {
  print("\(index): \(letter)")
}
```

for loops

```
let vehicles = ["unicycle" : 1, "bicycle" : 2, "tricycle" : 3, "quad bike" : 4]
for (vehicleName, wheelCount) in vehicles {
  print("A \(vehicleName) has \(wheelCount) wheels")
}
```

while loops

```
var numberOfLives = 3

while numberOfLives > 0 {
  playMove()
  updateLivesCount()
}
```

while loops

```
var numberOfLives = 3

while numberOfLives > 0 {
  print("I still have \(numberOfLives) lives.")
}
```

while loops

```
var numberOfLives = 3
var stillAlive = true

while stillAlive {
  print("I still have \(numberOfLives) lives.")
  numberOfLives -= 1
  if numberOfLives == 0 {
    stillAlive = false
  }
}
```

Control transfer statements

```
for counter in -10...10 {
  print(counter)
  if counter == 0 {
    break
  }
}
```

```
-10
-9
...
0
```

Swift Student Challenge

Apple is proud to support and uplift the next generation of developers, creators, and entrepreneurs with the Swift Student Challenge. The Challenge has given thousands of student developers the opportunity to showcase their creativity and coding capabilities through app playgrounds, and earn real-world skills that they can take into their careers and beyond. We're releasing new coding resources, working with community partners, and announcing the Challenge earlier than in previous years so students can dive deep into Swift and the development process — and educators can get a head start in supporting them.

Applications will open in February 2024 for three weeks.

New for the 2024 Challenge, out of 350 winners, we'll recognize 50 Distinguished Winners for their outstanding submissions and invite them to Apple in Cupertino for an extraordinary experience. All Challenge winners will receive one year of membership in the Apple Developer Program, a complimentary voucher to take an App Development with Swift certification exam, and a special gift from Apple.

Distinguished Winners

Exceptional work deserves exceptional recognition.

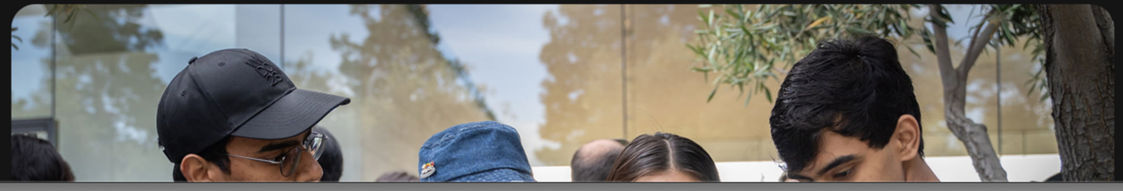
We're continually impressed by the outstanding submissions we receive for the Swift Student Challenge. And new for the 2024 Challenge, we're recognizing applicants who take their app playground concepts, design, and execution to the next level — with a next-level award.

Out of 350 total Challenge winners, we'll name 50 Distinguished Winners whose submissions demonstrate excellence in innovation, creativity, social impact, or inclusivity. This esteemed group will receive an invitation to join us — in person — next summer for three inspiring days at Apple in Cupertino, where they'll gain invaluable insights from Apple experts and engineers, connect with their peers through exciting activities, and enjoy a host of other engaging, unforgettable experiences. Travel and lodging included.

All Challenge winners will receive one year of membership in the Apple Developer Program, a complimentary voucher to take an App Development with Swift certification exam, and a special gift from Apple.

Stay tuned for more exciting details!

Get ready.



“Passion”

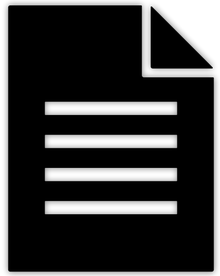
ALWAYS CREATE SOMETHING DIFFERENT

WWDC

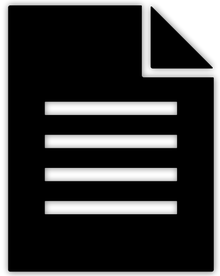
Swift Student Challenge

Your Submission includes

.swiftpm Application Playground



Beyond WWDC Essay



Application Technical Essay

WWDC

Swift Student Challenge

Application Technical Essay recommendations

WHO are you targeting?
WHAT is the purpose of the app?
WHEN what is the outcome of the app?
WHERE did you find the idea?
HOW does the app work?

Find a
WHY
for every question

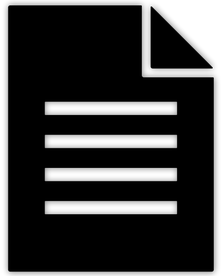
WWDC

Swift Student Challenge

Submissions will be judged on

Technical accomplishment

Creativity of ideas



Content of written responses

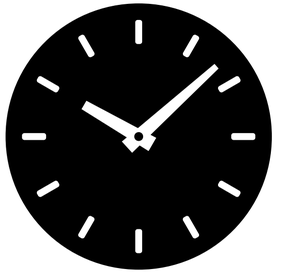
WWDC

Swift Student Challenge

My Recommendations

Be familiar with SwiftUI

Think of an idea prior to the competition



Time Management is key

Swift Coding Club

Inspire, Design, Code