Seus resultados

Você marcou 8/20. Veja abaixo um detalhamento de cada pergunta que você respondeu, incluindo explicações das respostas corretas.

Respostas a perguntas

Pergunta 1

Quando este código for executado, qual será o valor da constante `j`?

```
let i = "5"
let j = i + i
```

Sua resposta: Este código não será compilado.

Resposta correta: "55" (uma string).

Explanation: When used with strings, the **+** operator acts to append one string to another. In this case, it merges "5" onto "5" to make "55"

Pergunta 2

Qual saída será produzida pelo código abaixo?

```
let i = 3

switch i {
case 1:
    print("Number was 1")
case 2:
    print("Number was 2")
case 3:
    print("Number was 3")
}
```

Sua resposta: "O número era 3".

Resposta correta: Este código não será compilado.

Explicação: Swift exige que todas as instruções de switch sejam exaustivas. Este código não será compilado porque não tem uma cláusula **`default`**.

Pergunta 3

Qual saída será produzida pelo código abaixo?

```
for i in 1...3 {
    print(i)
}
```

Sua resposta: 2.

Resposta correta: 1, 2, 3.

Explicação: Isso usa o operador de intervalo fechado (...) para repetir de 1 a 3 inclusive.

Pergunta 4

Qual saída será produzida pelo código abaixo?

```
struct Spaceship {
    var name: String {
        willSet {
            print("I'm called \(newValue)!")
        }
    }
}

var serenity = Spaceship(name: "Serenity")
serenity.name = "TARDIS"
```

Você respondeu corretamente: "Eu sou chamado de TARDIS!".

Explanation: The `willSet` property observer is triggered only when the initial value is changed, and not when the struct is created using memberwise initialization.

Pergunta 5

Qual saída será produzida pelo código abaixo?

```
let oneMillion = 1_000_000
let oneThousand = oneMillion / 0_1_0_0_0
print(oneThousand)
```

Você respondeu corretamente: 1000.

Explicação: Swift permite que você use qualquer número de zeros iniciais antes de um número e qualquer número de sublinhados dentro de um número, a fim de facilitar a leitura. O exemplo dado, `0_1_0_0_0`, é improvável, mas uma maneira perfeitamente válida de escrever 1000.

Pergunta 6

Qual saída será produzida pelo código abaixo?

```
let names = ["Serenity", "Sulaco", "Enterprise", "Galactica"]

if let last = names.last {
    print(last)
}
```

Sua resposta: "Serenidade".

Resposta correta: "Galactica".

Explanation: Using `names.last` will return an optional string, but the `if let` unwraps that safely to produce "Galactica".

Pergunta 7

Qual saída será produzida pelo código abaixo?

```
var i = 2

repeat {
    i *= i * 2
} while i < 100

print(i)
```

Você respondeu corretamente: 128.

Explanation: Each time the loop goes around, the `i` is doubled then multiplied by itself. The first time through the loop it will be 8, and the second time it will be 128, at which point the loop will exit and print 128.

Pergunta 8

Qual saída será produzida pelo código abaixo?

```
for i in 3...1 {
    print(i)
}
```

Sua resposta: Nada será produzido.

Resposta correta: Este código será compilado, mas travará.

Explicação: Este código será compilado com sucesso, mas travará em tempo de execução: o Swift não permite que você gere intervalos em que o valor inicial seja maior que o valor final.

Pergunta 9

Quando este código for executado, o que ele imprimirá?

```
enum Direction: CaseIterable {
    case north, south, east, west
}

print(Direction.allCases.count)
```

Você respondeu corretamente: 4.

Explanation: In this code, the `CaseIterable` protocol is able to generate an `allCases` array that contains each case in our enum in the order it was defined.

Pergunta 10

Qual saída será produzida pelo código abaixo?

```
let names = ["Serenity", "Sulaco", "Enterprise", "Galactica"]

for name in names where name.hasPrefix("S") {
    print(name)
}
```

Você respondeu corretamente: "Serenidade", "Sulaco".

Explanation: The `where` condition for the loop will ensure that only names that start with the letter S will used inside the loop.

Pergunta 11

You are using a method named `willThrowAnError()`, and it is marked using `throws` because it will always throw the error "Forbidden". Given this context, what output will be produced by the code below?

```
do {
    try willThrowAnError()
} catch {
    print("The error message was: \(error)")
}
```

Sua resposta: Nada será produzido.

Resposta correta: "A mensagem de erro foi: Proibido".

Explanation: When in a `catch` block with no pattern, Swift automatically matches any error and binds it to a local constant called `error`.

Pergunta 12

Qual saída será produzida pelo código abaixo?

```
enum Weather {
    case sunny
    case cloudy
    case windy(speed: Int)
}

let today: Weather = .windy(speed: 10)

switch today {
case .sunny, .cloudy:
    print("It's not that windy")
case .windy(let speed) where speed >= 10:
    print("It's very windy")
default:
    print("It's a bit windy")
}
```

Você respondeu corretamente: "Está ventando muito".

Explanation: The `windy` case value has an associated value to store the wind speed as an integer. In the code, this is set to 10, which means the "It's very windy" case will be triggered in the `switch` block.

Pergunta 13

Qual saída será produzida pelo código abaixo?

```
let names = ["Chris", "Joe", "Doug", "Jordan"]

if let name = names[1] {
    print("Brought to you by \(name)")
}
```

Sua resposta: "Trazido a você por Joe".

Resposta correta: Este código não será compilado.

Explanation: Subscripting an array of strings will return a `String` rather than a `String?`, which means it is a compile error to attempt to unwrap it using `if let`.

Pergunta 14

When this code finishes executing, how many strings will the `names` array contain?

```
let names = [String]()
names.append("Amy")
names.append("Clara")
names.append("Rory")
```

Sua resposta: 2.

Resposta correta: Este código não será compilado.

Explanation: The `names` array was declared using `let`, which makes it a constant. This means it is a compile error to try to use `append()` to add strings to it.

Pergunta 15

Qual diretiva do compilador Swift forçará o compilador a emitir um erro?

Sua resposta: @error.

Resposta correta: #erro.

Explanation: The `#error` compiler directive forces Swift to issue an error. It's useful when you're sending code to someone and they need to fill in an important value, such as an API key.

Pergunta 16

Qual saída será produzida pelo código abaixo?

```
var motto = "Bow ties are cool"
motto.replacingOccurrences(of: "Bow", with: "Neck")
print(motto)
```

Sua resposta: "Bow".

Resposta correta: "Laços são legais".

Explanation: The `replacingOccurrences()` method returns a new string with its change effected, rather than modifying it in place. In this code, that return value is being discarded, so the original `motto` variable remains unchanged.

Pergunta 17

When this code is executed, what value will `num` have?

```
let num = UInt.min
```

Você respondeu corretamente: 0.

Explanation: `UInt` means "unsigned integer", which is a whole number that cannot be less than zero.

Pergunta 18

Qual saída será produzida pelo código abaixo?

```
let names = ["Amy", "Rory"]

for name in names {
    name = name.uppercased()
    print("HELLO, \(name)!")
}
```

Sua resposta: "OLÁ, RORY!".

Resposta correta: Este código não será compilado.

Explanation: This code will not compile because it modifies `name` inside the loop. If you want to do this, you must use the `var` keyword like this: `for var name in names`.

Pergunta 19

Qual saída será produzida pelo código abaixo?

```
var i = 2

do {
    print(i)
    i *= 2
} while (i < 128)
```

Sua resposta: 2, 4, 8, 16, 32, 64, 128.

Resposta correta: Este código não será compilado.

Explanation: The `do` keyword is invalid here; the programmer should use `repeat` instead.

Pergunta 20

Quando este código é executado, qual é o valor do `myStr`?

```
var myStr: String = "0"
myStr = "shiny"
```

Você respondeu corretamente: "brilhante".

Explicação: As constantes Swift não precisam receber um valor inicial, desde que recebam um valor apenas uma vez e antes de serem usadas.

Pronto para tentar novamente?

Se você quiser refaçar este teste ou tentar um dos outros, clique aqui.

Esta página foi útil? Deixe-nos saber!

☆☆☆☆☆

Clique aqui para visitar a loja Hacking with Swift >>

TWITTER Mastodon E-MAIL Patrocine o site

Sobre o Código do Glossário Política de Privacidade da Licença Política de Atualização da Política de Reembolso Código de Conduta

Você não está logado

Faça login ou crie uma conta