你能在 JavaScript 中停止"forEach"吗？

作者：微信公众号：【架构师老卢】
2-19 8:18

概述： 你能停止 JavaScript 中的 forEach 循环吗？这是我在一次面试中被问到的一个问题，我最初的回答是："不，我不能那样做。不幸的是，我的回答导致面试官突然结束了面试。我对结果感到沮丧，我问面试官："为什么？真的可以在 JavaScript 中停止 forEach 循环吗？在面试官回答之前，我花了一点时间解释我对为什么我们不能直接停止 JavaScript 中的 forEach 循环的理解。我的答案正确吗？ 朋友们，下面的代码会输出哪些数字？ 它会只输出一个数字还是多个数字？ 是的，它将输出 '0'， '1'， '2'， '3'。const array = [-3, -2, -1, 0,

你能停止 JavaScript 中的 forEach 循环吗？这是我在一次面试中被问到的一个问题，我最初的回答是："不，我不能那样做。

不幸的是，我的回答导致面试官突然结束了面试。

我对结果感到沮丧，我问面试官："为什么？真的可以在 JavaScript 中停止 forEach 循环吗？

在面试官回答之前，我花了一点时间解释我对为什么我们不能直接停止 JavaScript 中的 forEach 循环的理解。

我的答案正确吗？

朋友们，下面的代码会输出哪些数字？

它会只输出一个数字还是多个数字？

是的，它将输出 '0'， '1'， '2'， '3'。

```
const array = [ -3, -2, -1, 0, 1, 2, 3 ]

array.forEach((it) => {
  if (it >= 0) {
    console.log(it)
    return // or break
  }
})
```

```
> const array = [ -3, -2, -1, 0, 1, 2, 3 ]

  array.forEach((it) => {
    if (it >= 0) {
      console.log(it)
      return
    }
  })
0
1
2
3
```

没错！我把这段代码给面试官看，**但他仍然相信我们可以停止 JavaScript 中的 forEach 循环。**

为什么？

为了说服他，我不得不再次实施模拟。forEach

```
Array.prototype.forEach2 = function (callback, thisCtx) {
  if (typeof callback !== 'function') {
    throw `${callback} is not a function`
  }

  const length = this.length

  let i = 0
  while (i < length) {
    if (this.hasOwnProperty(i)) {
      // Note here. Each callback function will be executed once
      callback.call(thisCtx, this[ i ], i, this)
    }
    i++
  }
}
```

是的，当我们使用 'forEach' 遍历一个数组时，回调将对数组的每个元素执行一次。我们没有办法让早地脱离它。

例如，在以下代码中，即使"func1"遇到 break 语句，"2"仍将输出到控制台。

```
const func1 = () => {
  console.log(1)
  return
}

const func2 = () => {
  func1()
  console.log(2)
}
func2()
```

```
> const func1 = () => {
    console.log(1)
    return
  }

  const func2 = () => {
    func1()
    console.log(2)
  }

  func2()
  1
  2
```

停止 forEach 的 3 种方法

你很棒，但我想告诉你，**我们至少有 3 种方法可以在 JavaScript 中停止 forEach。**

1.# 抛出错误

找到第一个大于或等于 0 的数字后，此代码将无法继续。所以控制台只会打印出 0。

```
const array = [ -3, -2, -1, 0, 1, 2, 3 ]

try {
  array.forEach((it) => {
    if (it >= 0) {
      console.log(it)
      throw Error(`We've found the target element.`)
    }
  })
} catch (err) {

}
```

```
> const array = [ -3, -2, -1, 0, 1, 2, 3 ]

  try {
    array.forEach((it) => {
      if (it >= 0) {
        console.log(it)
        throw Error(`We've found the target element.`)
      }
    })
  } catch (err) {

  }
0
undefined
```

2.# 设置数组长度为 0

请不要那么惊讶，面试官对我说。

我们还可以通过将数组的长度设置为 0 来中断 forEach。如您所知，如果数组的长度为 0，则 forEach 不会执行任何回调。

```
const array = [ -3, -2, -1, 0, 1, 2, 3 ]

array.forEach((it) => {
  if (it >= 0) {
    console.log(it)
    array.length = 0
  }
})
```

```
> const array = [ -3, -2, -1, 0, 1, 2, 3 ]

  array.forEach((it) => {
    if (it >= 0) {
      console.log(it)
      array.length = 0
    }
  })
0
```

3.# 使用拼接删除数组的元素

思路与方法2相同，如果可以删除目标元素后面的所有值，则forEach将自动停止。

```
const array = [ -3, -2, -1, 0, 1, 2, 3 ]

array.forEach((it, i) => {
  if (it >= 0) {
    console.log(it)
    // Notice the sinful line of code
    array.splice(i + 1, array.length - i)
  }
})
```

```
> const array = [ -3, -2, -1, 0, 1, 2, 3 ]

  array.forEach((it, i) => {
    if (it >= 0) {
      console.log(it)
      // Notice the sinful line of code
      array.splice(i + 1, array.length - i)
    }
  })
0
```

我瞎大了眼睛，我不想读这段代码。这很糟糕。

请使用或一些

我对面试官说。"哦，也许你是对的，你说法在 JavaScript 中停止了 forEach，但我认为你的老板会解雇你，因为这是一个非常糟糕的代码片段。

我不喜欢做这样的事情,这会让我的同事讨厌我。

也许我们应该使用"for"或"some"方法来解决这个问题。

1. for

```
const array = [ -3, -2, -1, 0, 1, 2, 3 ]

for (let i = 0, len = array.length; i < len; i++) {
  if (array[ i ] >= 0) {
    console.log(array[ i ])
    break
  }
}
```

```
}
```

```
const array = [ -3, -2, -1, 0, 1, 2, 3 ]

for (let i = 0, len = array.length; i < len; i++) {
  if (array[ i ] >= 0) {
    console.log(array[ i ])
    break
  }
}
```

```
0
```

2. some

```
const array = [ -3, -2, -1, 0, 1, 2, 3 ]


array.some((it, i) => {
  if (it >= 0) {
    console.log(it)
    return true
  }
})
```

```
const array = [ -3, -2, -1, 0, 1, 2, 3 ]

array.some((it, i) => {
  if (it >= 0) {
    console.log(it)
    return true
  }
})
```

```
0
```