1

Ray Fixは誰？

Source: https://en.wikipedia.org/wiki/La_Jolla

2

RayWenderlich.com

3

ECHO
LABORATORIES

4-1

ECHO
LABORATORIES

4-2

5

スタックvsヒープ領域

6

スタック

スタックはとにかく速い！

7-1

スタック

スタックはとにかく速い！

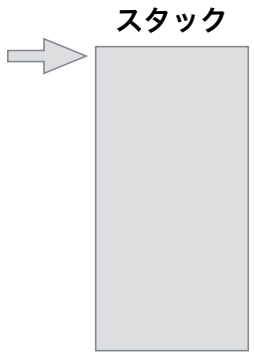
```
func candy() {

  let ramune = 10
  let pocky = 12
  let gummi = 20

}
```


7-2

スタック

スタックはとにかく速い！

```
func candy() {

  let ramune = 10
  let pocky = 12
  let gummi = 20

}
```

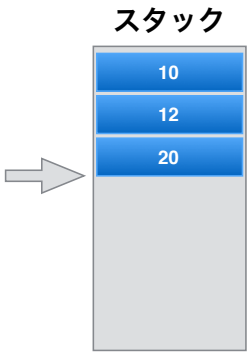


7-3

スタック

スタックはとにかく速い！

```
func candy() {

  let ramune = 10
  let pocky = 12
  let gummi = 20

}
```

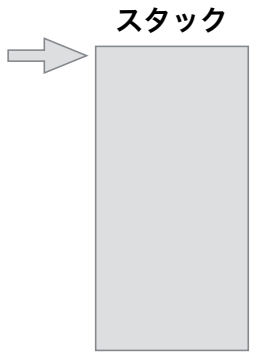


7-4

スタック

スタックはとにかく速い！

```
func candy() {

  let ramune = 10
  let pocky = 12
  let gummi = 20

}
```

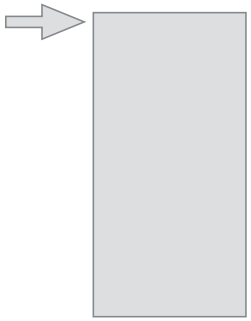


7-5

スタック

スタックはとにかく速い！

スタック



```
func candy() {

  let ramune = 10
  let pocky = 12
  let gummi = 20

}
```

ロックする必要は全くない

ヒープ領域

```
class Dish  {

    var name: String

    init(name: String) {
        self.name = name
        print("🍻 Dish \(name)")
    }

    deinit {
        print("☠️ Dish \(name)")
    }
}
```

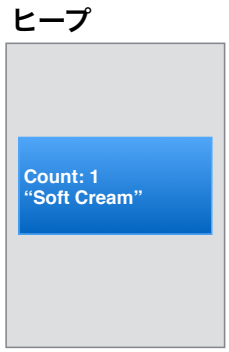

ヒープ領域

ヒープ領域

```
let softCream = Dish(name: “Soft Cream")
```

ヒープ領域

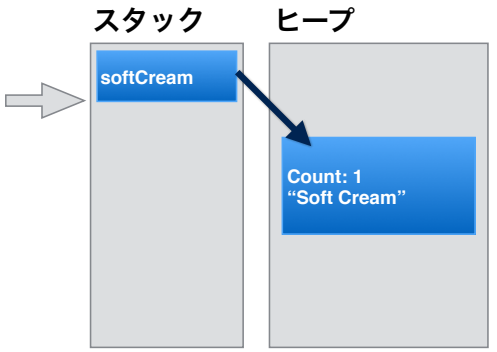
```
let softCream = Dish(name: “Soft Cream")
```



9-3

ヒープ領域

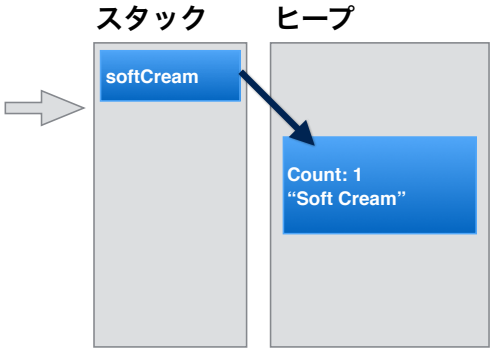
```
let softCream = Dish(name: “Soft Cream")
```



9-4

ヒープ領域

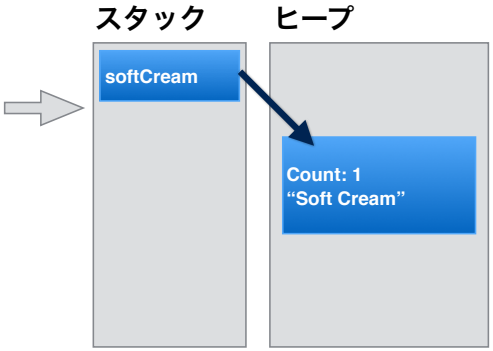
```
let softCream = Dish(name: “Soft Cream")
```



10-1

ヒープ領域

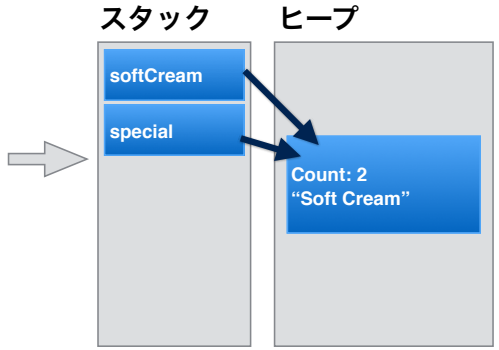
```
let softCream = Dish(name: “Soft Cream")
let special = softCream
```



10-2

ヒープ領域

```
let softCream = Dish(name: “Soft Cream")
let special = softCream
```



10-3

共有

```
softCream.name = "Choco Soft Cream"
special.name  // "Choco Soft Cream"
```


11-1

共有

```
softCream.name = "Choco Soft Cream"
special.name  // "Choco Soft Cream"
```

ワオ、便利！！！

11-2

共有

```
softCream.name = "Choco Soft Cream"
special.name  // "Choco Soft Cream"
```

ワオ、便利！！！

しかし

11-3

共有

```
softCream.name = "Choco Soft Cream"
special.name  // "Choco Soft Cream"
```

ワオ、便利！！！

しかし

```
special.name = "ピーマン"
softCream.name // "ピーマン"
```


共有

```
softCream.name = "Choco Soft Cream"
special.name  // "Choco Soft Cream"
```

ワオ、便利！！！

しかし

```
special.name = "ピーマン"
softCream.name // "ピーマン"
```

想定外の共有は参照型の欠点

解決法:定数を使う

```
class Dish  {

    var name: String

    init(name: String) {
        self.name = name
        print("🍛 Dish \(name)")
    }

    deinit {
        print("☠ Dish \(name)")
    }
}
```


解決法:定数を使う

```
class Dish  {

    var name: String

    init(name: String) {
        self.name = name
        print("🍛 Dish \(name)")
    }

    deinit {
        print("☠ Dish \(name)")
    }
}
```

解決法:定数を使う

```
class Dish  {

    let name: String

    init(name: String) {
        self.name = name
        print("🐤 Dish \(name)")
    }

    deinit {
        print("☠️ Dish \(name)")
    }
}
```

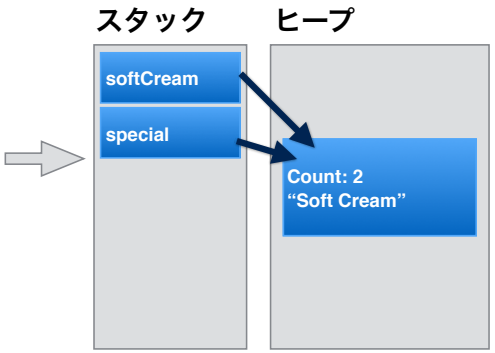

解決法:定数を使う

```
class Dish  {

    let name: String

    init(name: String) {
        self.name = name
        print("🐤 Dish \(name)")
    }

    deinit {
        print("☠️ Dish \(name)")
    }
}
```

⊗ `special.name = "ピーマン"`

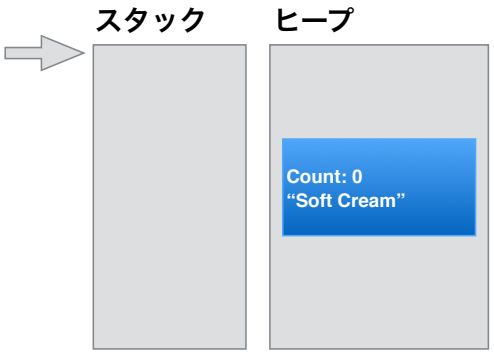
ヒープ領域

```
let softCream = Dish(name: “Soft Cream")
let special = softCream
```



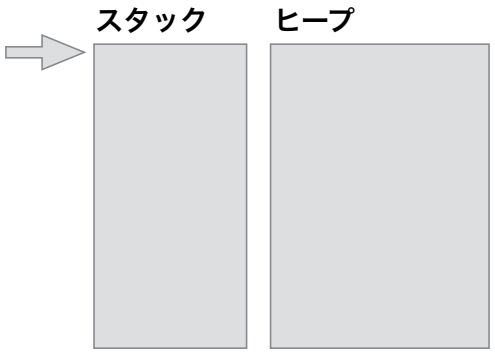
ヒープ領域

```
let softCream = Dish(name: “Soft Cream")
let special = softCream
```



ヒープ領域

```
let softCream = Dish(name: “Soft Cream")
let special = softCream
```



問題：循環参照

参照サイクル

```
class Customer {
  var orders: [Order]

  func add(order: Order) {
    order.customer = self
    orders.append(order)
  }
}


class Order {
  var customer: Customer?
  let dish: Dish
}
```


参照サイクル

```
class Customer {
  var orders: [Order]

  func add(order: Order) {
    order.customer = self
    orders.append(order)
  }
}


class Order {
  var customer: Customer?
  let dish: Dish
}
```

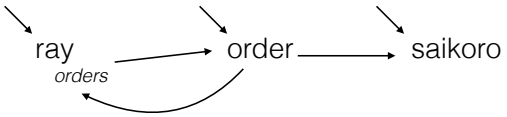
参照サイクル

```
let ray = Customer(name: "Ray")
let saikoro = Dish(name: “Saikoro Steak")
let order = Order(dish: saikoro)
ray.add(order: order)
```


16-1

参照サイクル

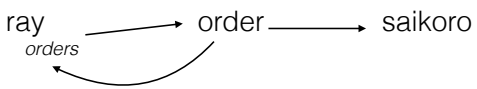
```
let ray = Customer(name: "Ray")
let saikoro = Dish(name: “Saikoro Steak")
let order = Order(dish: saikoro)
ray.add(order: order)
```



16-2

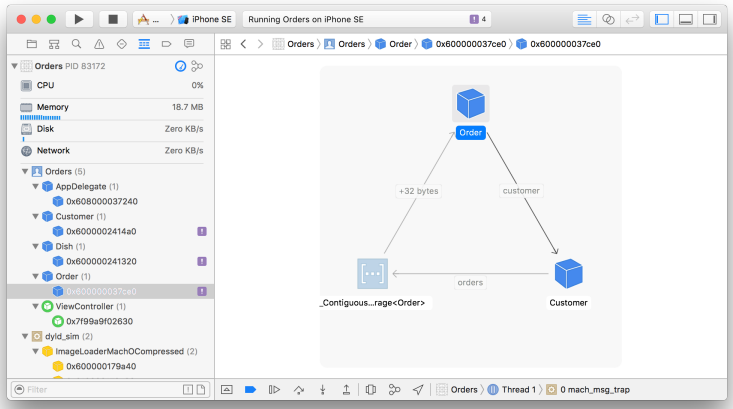
参照サイクル

```
let ray = Customer(name: "Ray")
let saikoro = Dish(name: “Saikoro Steak")
let order = Order(dish: saikoro)
ray.add(order: order)
```



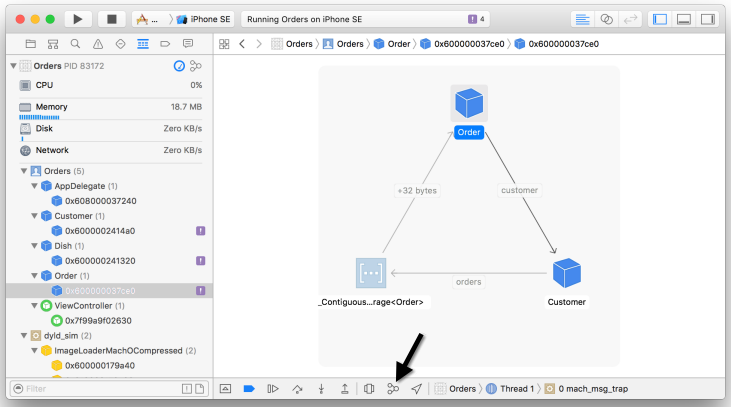
16-3

Xcode 8 Memory Visualizer



17-1

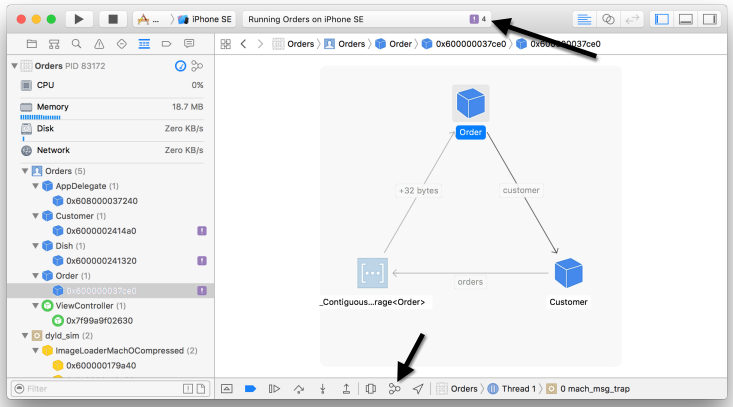
Xcode 8 Memory Visualizer



iosdc.jp Ray Fix

17-2

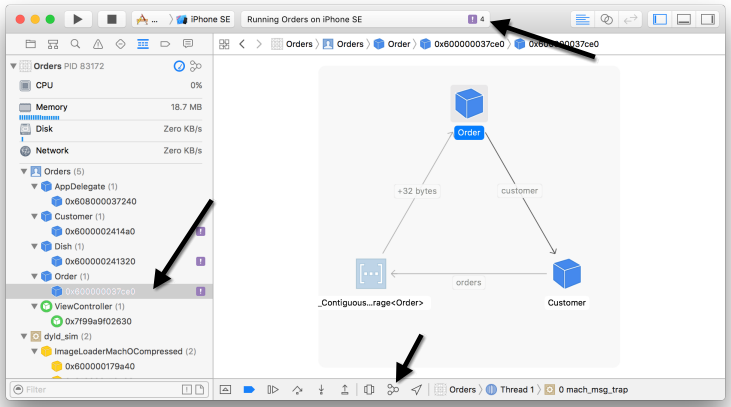
Xcode 8 Memory Visualizer



iosdc.jp Ray Fix

17-3

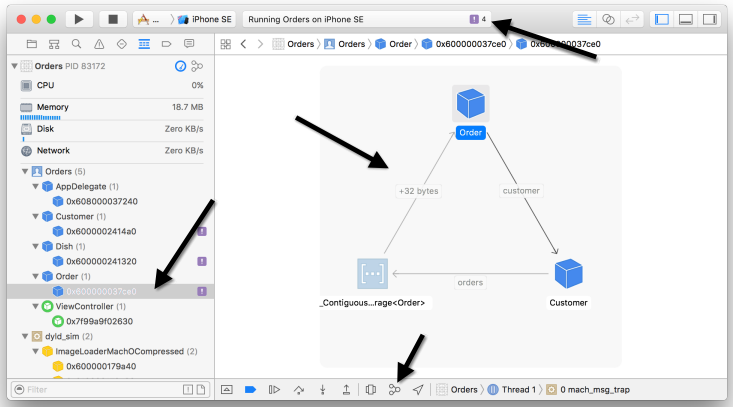
Xcode 8 Memory Visualizer



iosdc.jp Ray Fix

17-4

Xcode 8 Memory Visualizer



iosdc.jp Ray Fix

17-5

循環参照

18

循環参照

```
class Customer {
  var orders: [Order]

  func add(order: Order) {
    order.customer = self
    orders.append(order)
  }
}


class Order {
  var customer: Customer?
  let dish: Dish
}
```


19-1

循環参照

```
class Customer {
  var orders: [Order]

  func add(order: Order) {
    order.customer = self
    orders.append(order)
  }
}


class Order {
  weak var customer: Customer?
  let dish: Dish
}
```


19-2

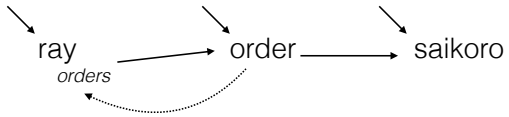
循環参照

```
class Customer {
  var orders: [Order]

  func add(order: Order) {
    order.customer = self
    orders.append(order)
  }
}


class Order {
  weak var customer: Customer?
  let dish: Dish
}
```


19-3

循環参照

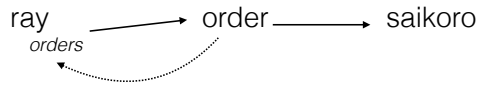
```
let ray = Customer(name: "Ray")
let saikoro = Dish(name: "Saikoro")
let order = Order(dish: saikoro)
ray.add(order: order)
```



20-1

循環参照

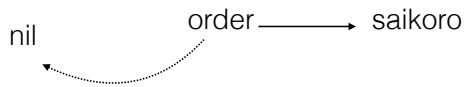
```
let ray = Customer(name: "Ray")
let saikoro = Dish(name: "Saikoro")
let order = Order(dish: saikoro)
ray.add(order: order)
```



20-2

循環参照

```
let ray = Customer(name: "Ray")
let saikoro = Dish(name: "Saikoro")
let order = Order(dish: saikoro)
ray.add(order: order)
```



20-3

循環参照

```
let ray = Customer(name: "Ray")
let saikoro = Dish(name: "Saikoro")
let order = Order(dish: saikoro)
ray.add(order: order)
```

saikoro

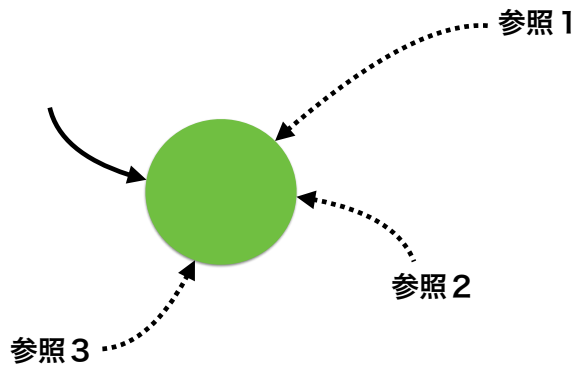
20-4

循環参照

```
let ray = Customer(name: "Ray")
let saikoro = Dish(name: "Saikoro")
let order = Order(dish: saikoro)
ray.add(order: order)
```

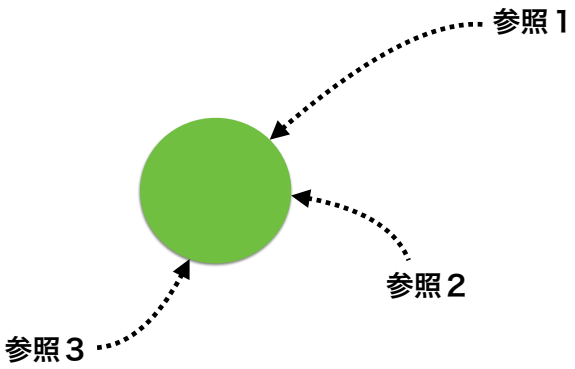

20-5

weak



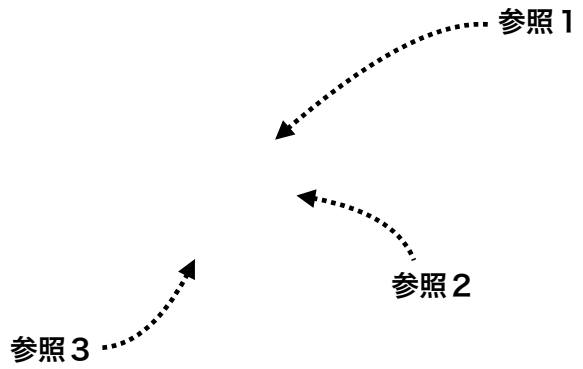
21-1

weak

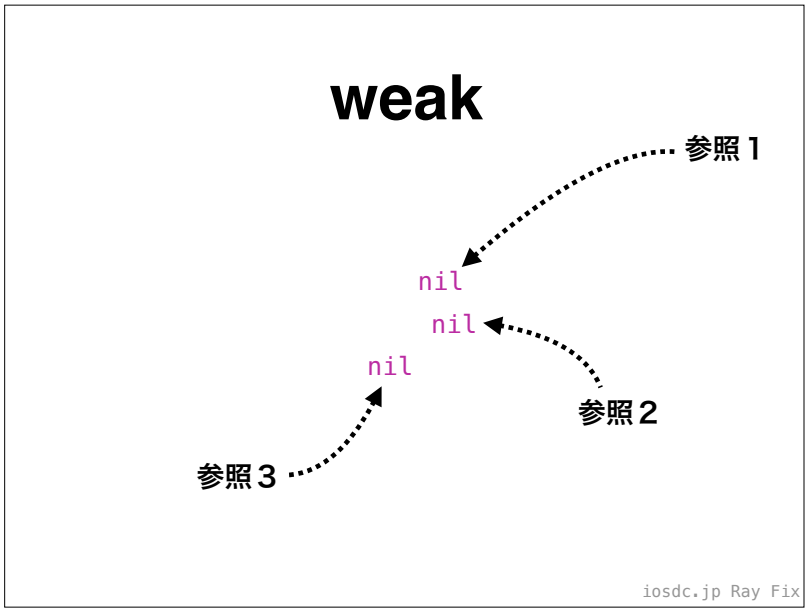


21-2

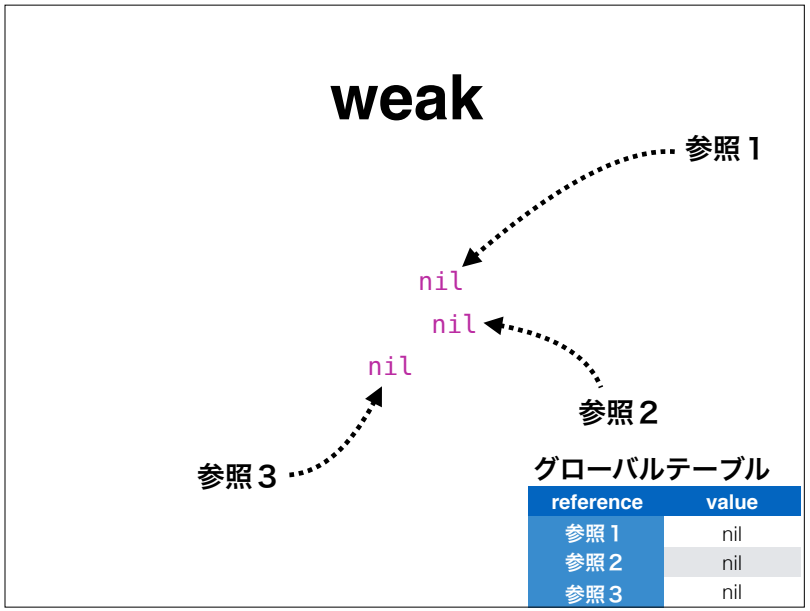
weak



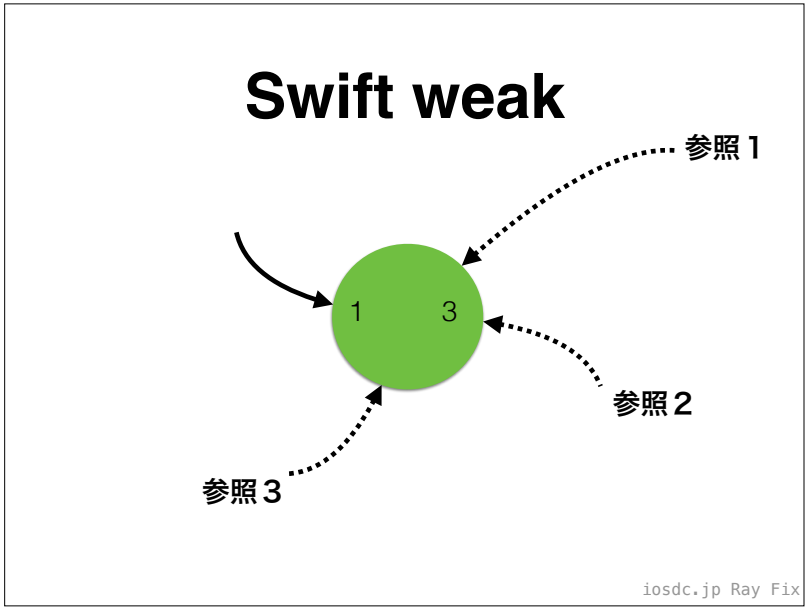
21-3



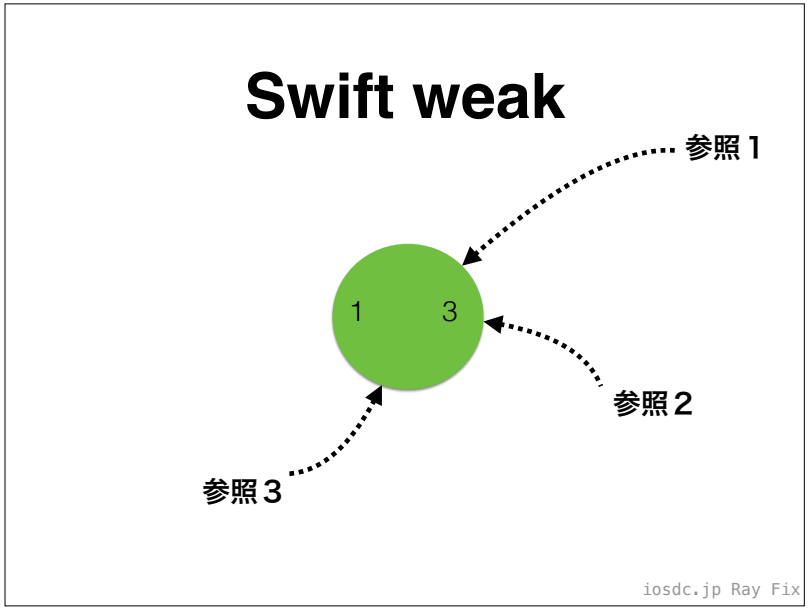
21-4



21-5

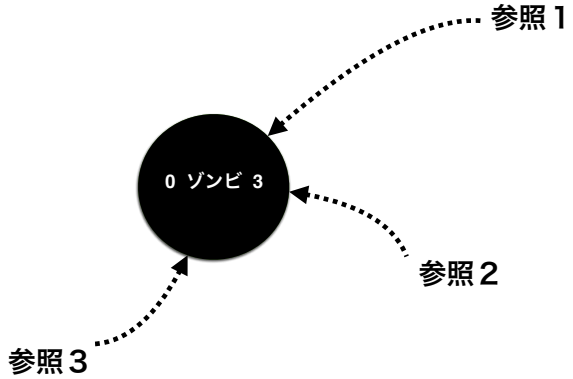


22-1



22-2

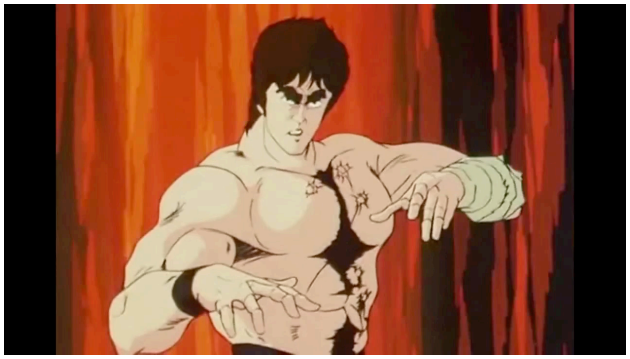
Swift weak



iosdc.jp Ray Fix

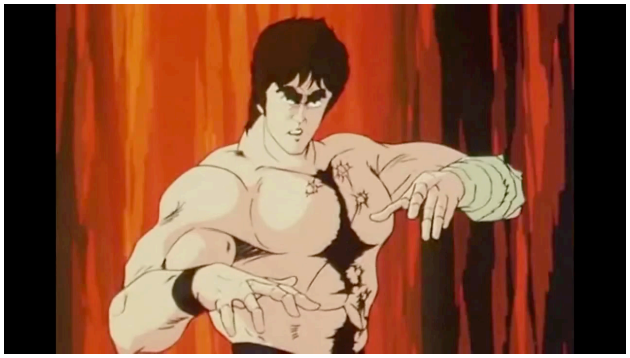
22-3

強い参照カウントが０に



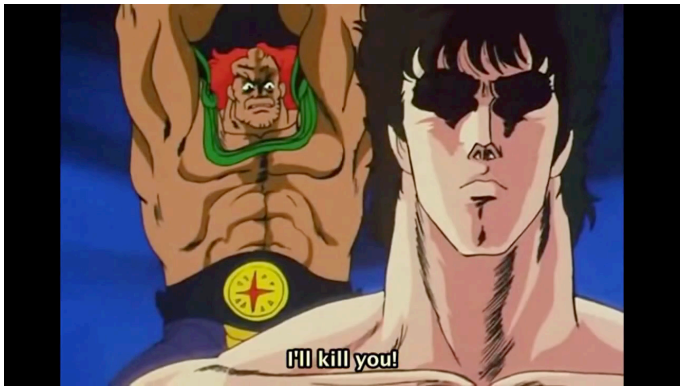
23-1

強い参照カウントが０に



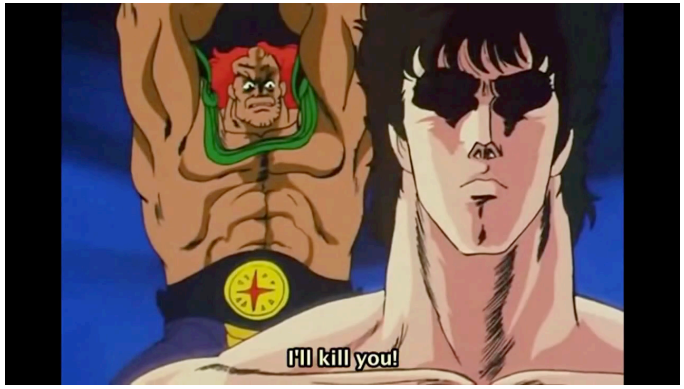
23-2

弱い参照カウントが０に



24-1

弱い参照カウントが０に



unowned

- unowned は別の種類の弱い参照
- 参照先が必ずあることが前提
- もし参照先がないと、プログラムが停止する

unowned

```
class Customer {
 var orders: [Order]

  func add(dish: Dish) {
    let order = Order(dish: dish, customer: self)
    orders.append(order)
  }
}


class Order {
   let customer: Customer
   let dish: Dish
}
```


unowned

```
class Customer {
 var orders: [Order]

  func add(dish: Dish) {
    let order = Order(dish: dish, customer: self)
    orders.append(order)
  }
}


class Order {
          let customer: Customer
   let dish: Dish
}
```

unowned

```
class Customer {
 var orders: [Order]

  func add(dish: Dish) {
    let order = Order(dish: dish, customer: self)
    orders.append(order)
  }
}


class Order {
  unowned let customer: Customer
  let dish: Dish
}
```

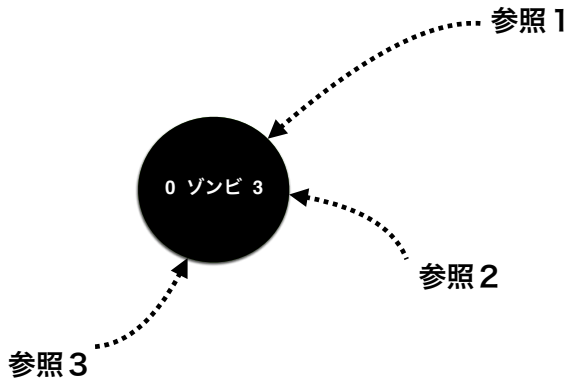
26-3

unowned

```
class Customer {
 var orders: [Order]

  func add(dish: Dish) {
    let order = Order(dish: dish, customer: self)
    orders.append(order)
  }
}


class Order {
  unowned let customer: Customer
  let dish: Dish
}
```

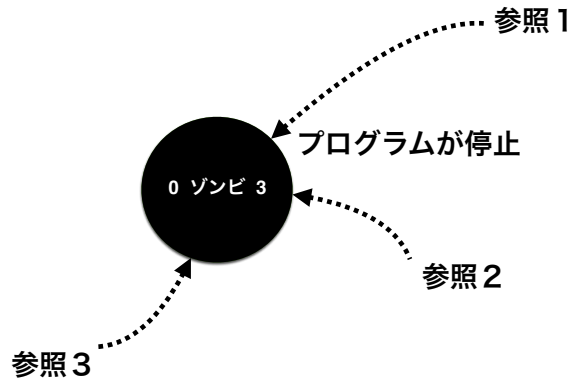
26-4

unowned



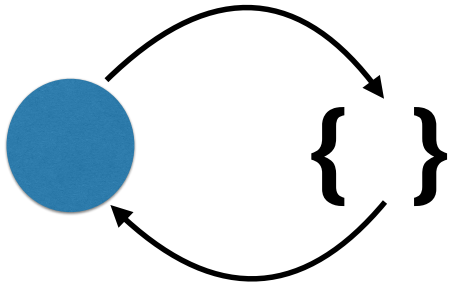
27-1

unowned



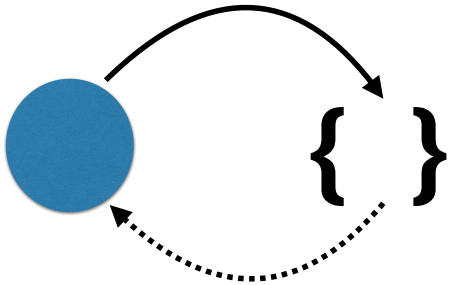
27-2

関数とクロージャも参照型



28-1

関数とクロージャも参照型



28-2

例：寿司屋

```
enum MenuItem: String {
  case toro, ebi, anago, uni, ikura, hamachi
}

typealias Action = ()->()
```

29

例： 寿司屋

```
class Sushiya {
  lazy var menu: [MenuItem: Action] = […]

  func prepare(_ menuItem: MenuItem) {
    menu[menuItem]?()
  }

  private func serve(dish: Dish) {
    print("Now serving \(dish.name)")
  }
}
```

30

例： 寿司屋

```
menu =

[
    .toro: {
      let dish = Dish(name: "Toro")
      serve(dish: dish)
    },

    …
]
```

31-1

例： 寿司屋

```
menu =

[
    .toro: {
      let dish = Dish(name: "Toro")
      self.serve(dish: dish)
    },

    …
]
```

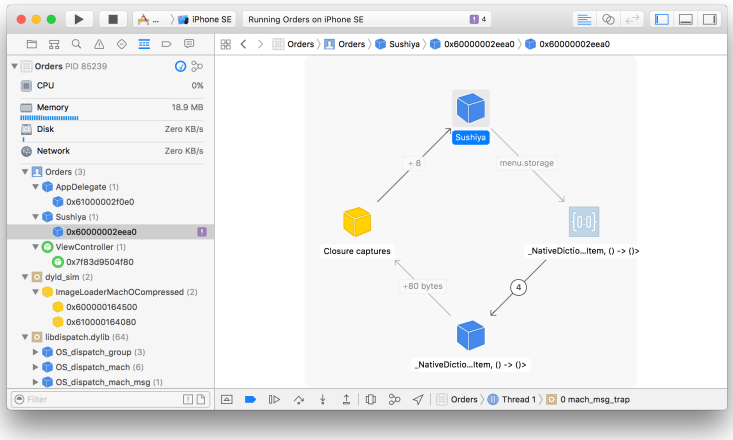
31-2

例： 寿司屋

```
menu =

[
    .toro: {
      let dish = Dish(name: "Toro")
      self.serve(dish: dish)
    },

    …
]
                let sushiya = Sushiya()
                sushiya.prepare(.toro)
```

31-3

メモリー漏れ



32

解決法： キャプチャーリスト

```
var value = 0

let showValue = {
  print(value)
}
```


解決法： キャプチャーリスト

```
var value = 0

let showValue = {
  print(value)
}


showValue()  // prints 0
value = 10
showValue()  // prints 10
```


解決法： キャプチャーリスト

```
var value = 0

let showValue = {
  print(value)
}


showValue()  // prints 0
value = 10
showValue()  // prints 10
```


解決法： キャプチャーリスト

```
var value = 0

let showValue = { [value] in
  print(value)
}


showValue()  // prints 0
value = 10
showValue()  // prints 10
```

解決法： キャプチャーリスト

```
var value = 0

let showValue = { [value] in
  print(value)
}


showValue()  // prints 0
value = 10
showValue()  // prints 0
```

34-3

解決法： キャプチャーリスト

```
menu =

[
    .toro: {
      let dish = Dish(name: "Toro")
      self.serve(dish: dish)
    },

    …
]
```

35-1

解決法： キャプチャーリスト

```
menu =

[
    .toro: { [unowned self] in
      let dish = Dish(name: "Toro")
      self.serve(dish: dish)
    },

    …
]
```

35-2

非同期の問題

```
private func serve(dish: Dish) {

   print("Now serving \(dish.name)")


}
```

36-1

非同期の問題

```
private func serve(dish: Dish) {
  DispatchQueue.main.async {
   print("Now serving \(dish.name)")
   self.served += 1
  }
}
```


非同期の問題

```
private func serve(dish: Dish) {
  DispatchQueue.main.async { [unowned self] in
   print("Now serving \(dish.name)")
   self.served += 1
  }
}
```


非同期の問題

```
private func serve(dish: Dish) {
  DispatchQueue.main.async { [unowned self] in
   print("Now serving \(dish.name)")
   self.served += 1
  }
}
```

⊗ **CRASH!!!!**

非同期の問題

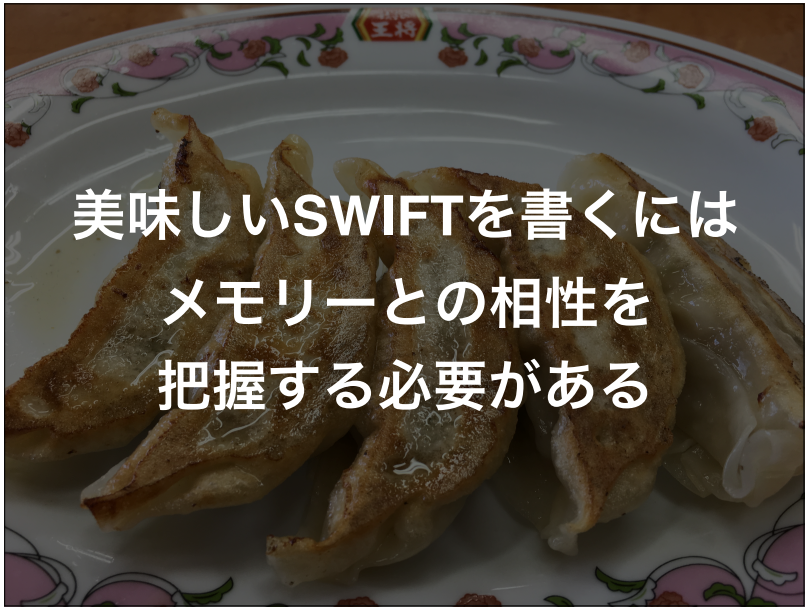
```
private func serve(dish: Dish) {
  DispatchQueue.main.async {
   print("Now serving \(dish.name)")
   self.served += 1
  }
}
```

weak

```
private func serve(dish: Dish) {
  DispatchQueue.main.async { [weak self] in
    print("Now serving \(dish.name)")
    self?.served += 1
  }
}
```


strong weak ダンス

```
private func serve(dish: Dish) {
  DispatchQueue.main.async { [weak self] in
    guard let strongSelf = self else {
      print("Cancelled \(dish.name)")
      return
    }
    print("Now serving \(dish.name)")
    strongSelf.served += 1
  }
}
```

ご清聴ありがとう
ございました

参考リンク

Ray Wenderlich メモリーチュート (Maxime Defauw著者) 9月公開予定
http://raywenderlich.com/

ECHO LABS 顕微鏡
http://echo-labs.com/

SWIFT 弱い参照の詳しい話
https://www.mikeash.com/pyblog/friday-qa-2015-12-11-swift-weak-references.html

横浜リナックス読書会６７：malloc
https://www.youtube.com/watch?v=0-vWT-t0UHg

寿司の写真
https://ja.wikipedia.org/wiki/寿司

WWDC Swift Performance
wwdc2016/416/

南カリフォルニアの写真
https://en.wikipedia.org/wiki/La_Jolla

ソースコード、スライド
https://github.com/rayfix/MemoryDish

友達の LINE スタムプ！
http://bit.ly/etystamp

41-1

参考リンク

Ray Wenderlich メモリーチュート (Maxime Defauw著者) 9月公開予定
http://raywenderlich.com/

ECHO LABS 顕微鏡
http://echo-labs.com/

SWIFT 弱い参照の詳しい話
https://www.mikeash.com/pyblog/friday-qa-2015-12-11-swift-weak-references.html

横浜リナックス読書会６７：malloc
https://www.youtube.com/watch?v=0-vWT-t0UHg

寿司の写真
https://ja.wikipedia.org/wiki/寿司

WWDC Swift Performance
wwdc2016/416/

南カリフォルニアの写真
https://en.wikipedia.org/wiki/La_Jolla

ソースコード、スライド
https://github.com/rayfix/MemoryDish

友達の LINE スタムプ！
http://bit.ly/etystamp

41-2