Scala Training 5

Recap + common functions

Letgo - 17/03/2017

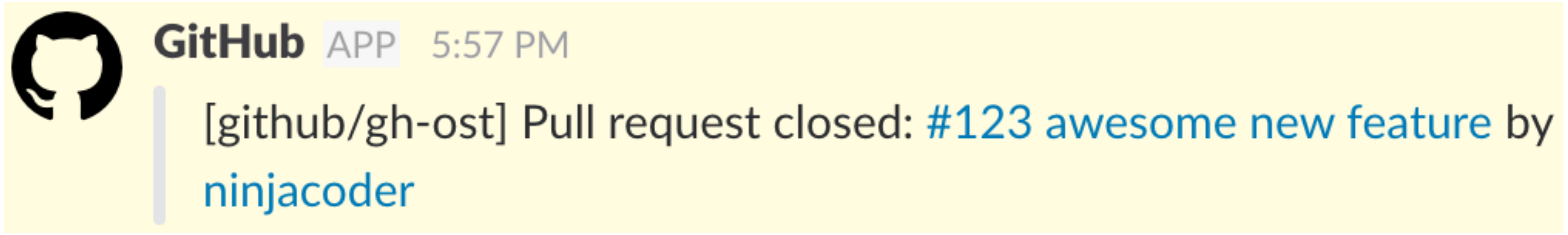
@gmadorell
@JavierCane

Contents

Session goal

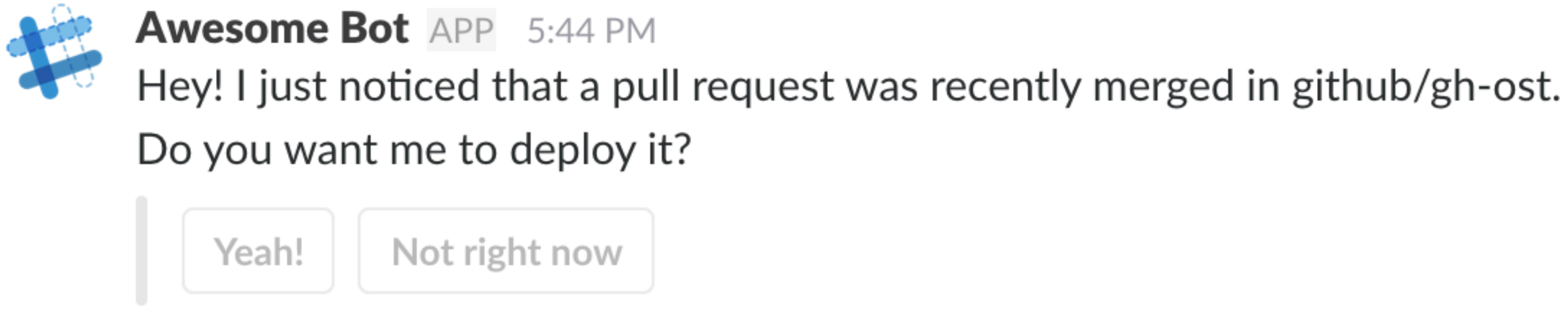
Session goal

- Hands on recap
- Use case:
  - Input:


GitHub APP 5:57 PM

[github/gh-ost] Pull request closed: #123 awesome new feature by ninjacoder

  - Output:


Awesome Bot APP 5:44 PM

Hey! I just noticed that a pull request was recently merged in github/gh-ost. Do you want me to deploy it?

Recap

Variables

```scala
case class Message(text: String)

var questionMessage: Message = Message("Ola k ase?”)

questionMessage = Message(“No ase nada?")
```

Values

```scala
val helloMessage = Message("Hello")

helloMessage = Message(“Bye") // error: Reassignment to val

val randomMessageAsVal: Message = allMessages(Random.nextInt(allMessages.size))
```

Functions

```scala
def randomMessageAsDef: Message = allMessages(Random.nextInt(allMessages.size))
```

Classes

```scala
class User(name: String, age: Int) {
  def getName: String = name
  def getAge: Int = age
}
```

```scala
val user =
  new User("lele", 18)

user.getName
```

Classes + val

```scala
class UserWithVals(
  val name: String,
  val age: Int
)
```

```scala
val userWithVals =
  new UserWithVals("lele", 18)

userWithVals.name
```

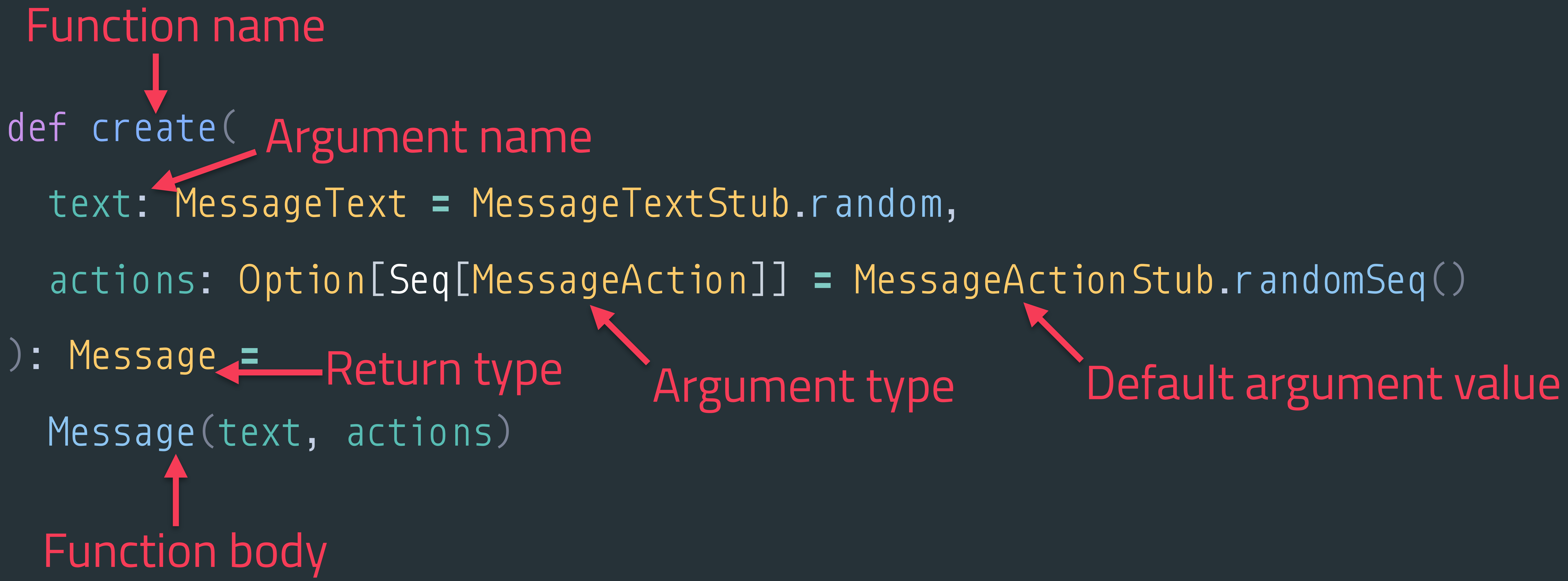
Case classes

```scala
case class UserCaseClass(
  name: String,
  age: Int
)
```

```scala
val userCaseClass =
  UserCaseClass("lele", 18)

userCaseClass.name
```

equals
copy
deconstruction

Syntax - Functions


```
def create(
  text: MessageText = MessageTextStub.random,
  actions: Option[Seq[MessageAction]] = MessageActionStub.randomSeq()
): Message =
  Message(text, actions)
```

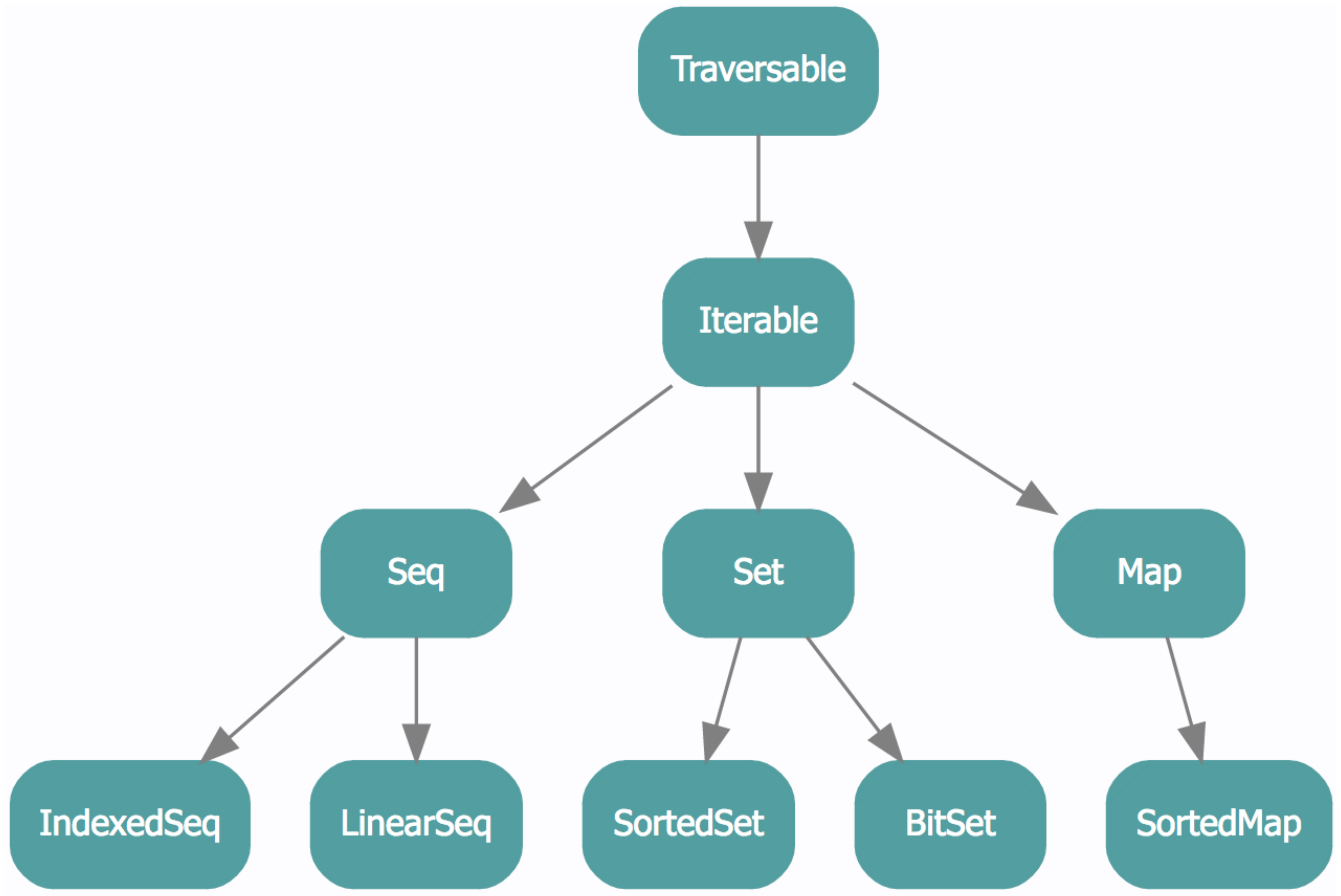
Higher order Functions

```scala
def increase(number: Int): Int = number + 1

val three = increase(2)


val numbers = Seq(1, 2, 3)

val increasedNumbers = numbers.map(number ⇒ increase(number))


val increasedNumbers2 = numbers.map(increase)
```

Testing

```
final class ListTest extends WordSpec with Matchers {
  "A List" should {
    "have a size of zero when it's created as empty" in {
      val emptyList: List[Int] = List.empty[Int]

      emptyList.size shouldBe 0
    }
  }
}
```

Collections



💡Bonus tracks:

- http://www.decodified.com/scala/collections-api.xml
- http://docs.scala-lang.org/overviews/collections/performance-characteristics.html
- Collections decission tree

Common functions

Imperative example

```
def isOdd(n: Int): Boolean = n % 2 ≠ 0

def square(n: Int): Int = n * n


def squaredOddNumbersUpToImperative(n: Int): Seq[Int] = {
  var sequence = mutable.Buffer[Int]()
  for (i ← 0 to n) {
    if (isOdd(i)) {
      sequence :+= square(i)
    }
  }
  sequence
}
```

Imperative example

```
def isOdd(n: Int): Boolean = n % 2 ≠ 0

def square(n: Int): Int = n * n


def squaredOddNumbersUpToImperative(n: Int): Seq[Int] = {
  var sequence = mutable.Buffer[Int]()
  for (i ← 0 to n) {
    if (isOdd(i)) {
      sequence :+= square(i)
    }
  }
  sequence
}
```

Example with filter

```scala
def isOdd(n: Int): Boolean = n % 2 ≠ 0

def square(n: Int): Int = n * n


def squaredOddNumbersUpToFilter(n: Int): Seq[Int] = {
  var sequence = mutable.Buffer[Int]()
  for (i ← (0 to n).filter(isOdd)) {
    sequence :+= square(i)
  }
  sequence
}
```

Example with filter

```
def isOdd(n: Int): Boolean = n % 2 ≠ 0

def square(n: Int): Int = n * n


def squaredOddNumbersUpToFilter(n: Int): Seq[Int] = {
  var sequence = mutable.Buffer[Int]()
  for (i ← (0 to n).filter(isOdd)) {
    sequence :+= square(i)
  }
  sequence
}
```

Example with filter

```scala
def isOdd(n: Int): Boolean = n % 2 ≠ 0

def square(n: Int): Int = n * n


def squaredOddNumbersUpToFilter(n: Int): Seq[Int] = {
  var sequence = mutable.Buffer[Int]()
  for (i ← (0 to n).filter(isOdd)) {
    sequence :+= square(i)
  }
  sequence
}
```

Example with filter and map

```
def isOdd(n: Int): Boolean = n % 2 ≠ 0

def square(n: Int): Int = n * n


def squaredOddNumbersUpToFunctional(n: Int): Seq[Int] =
  (0 to n).filter(isOdd).map(square)
```

Example with filter and map

```
def isOdd(n: Int): Boolean = n % 2 ≠ 0

def square(n: Int): Int = n * n


def squaredOddNumbersUpToFunctional(n: Int): Seq[Int] =
  (0 to n).filter(isOdd).map(square)
```

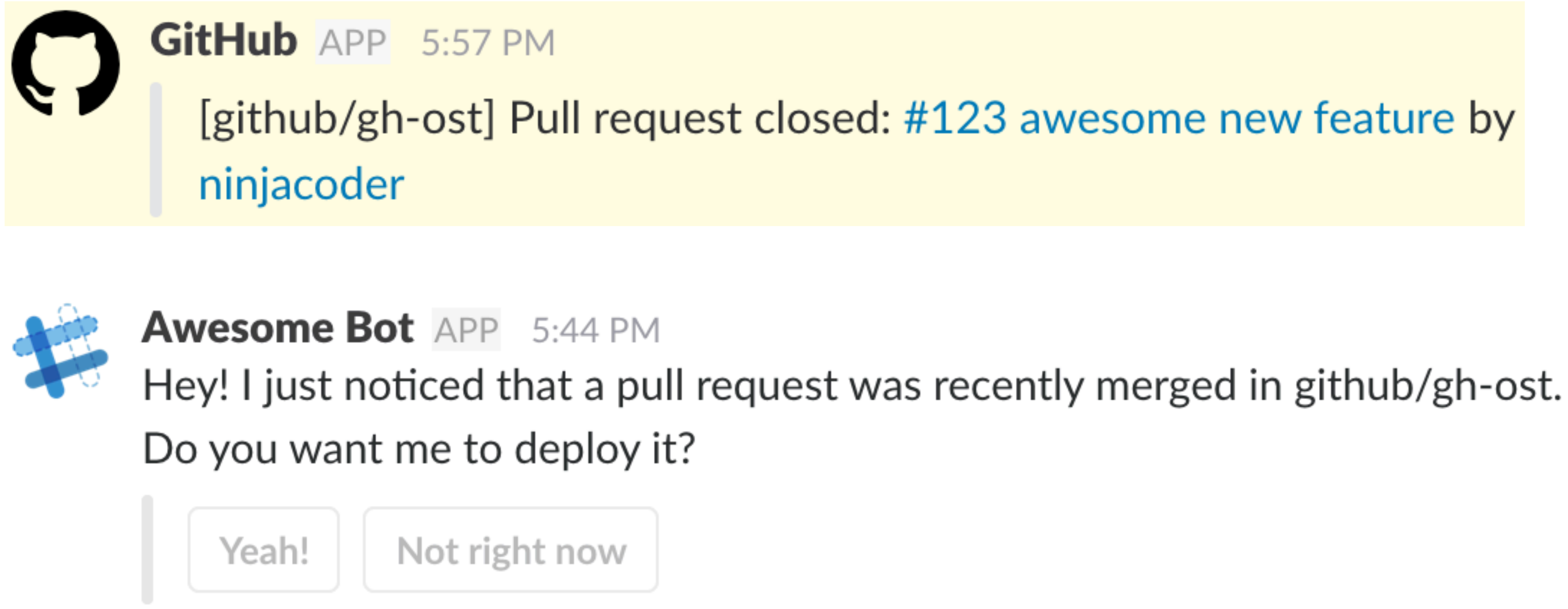
Common functions

- Scala is an OOP => Functions included in traits
- Common functions are consistent across the standard API “containers”
  - `map`
    - `scala.collection.TraversableLike.WithFilter#map`
    - `scala.concurrent.Future#map`
    - `scala.Option#map`
  - More examples: `flatMap`, `foreach`, `fold`, `filter`, `exists`, `contains`

Workshop

Workshop

- https://github.com/letgoapp/scala_course/tree/master/doc/lessons
- Implement the missing parts in order make the test pass


GitHub APP 5:57 PM

[github/gh-ost] Pull request closed: #123 awesome new feature by ninjacoder

Awesome Bot APP 5:44 PM

Hey! I just noticed that a pull request was recently merged in github/gh-ost. Do you want me to deploy it?

Workshop

- Implement an use case. Based on some input, send a message. I.E.:
  - Ask for the priority level when someone says your name
  - Suggest specific persons to contact to when someones says the name of your team
  - Interact with an external API in order to retrieve some information based on some message
  - ...