

Creating Your Control Structures



Harit Himanshu

@harittweets



Overview



Using function value to remove code duplication

Understanding currying

Creating control abstraction

Understanding *by-name* parameters

Project demo



Function Common Parts

```
def getHighPriorityTasks = {
```

```
  for (task <- tasks; if task.priority == high)
    yield task
```

```
}
```

Uncommon

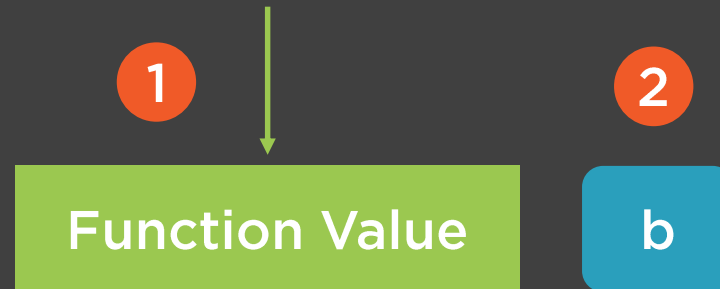
Common



Curry Function Invocation

```
def multiplyCurry(a: Int)(b: Int) = a * b
```

```
multiplyCurry: (a: Int)(b: Int)Int
```



if and for Loops

```
if (someCondition) {  
    // do something  
}
```

```
for (someCondition) {  
    // do something  
}
```



Rules for Creating Control Structure

Curry Up

2

```
def time(n: Int)(operation: Int => Unit): Unit = {  
  // implementation  
}
```

1

Function Literal with
One Argument



By-name Parameters

`() => Boolean`

FROM

`=> Boolean`

TO



By-name Syntax Alternative?

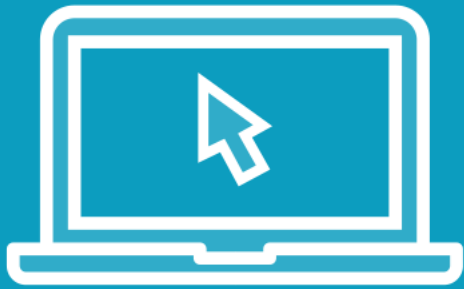
```
def assertTrue(predicate: => Boolean): Boolean = predicate
```

OR

```
def assertTrue(predicate: Boolean): Boolean = predicate
```



Demo



Refactor **BankOfScala**

Abstract Account opening process



Summary



Using function value to remove code duplication

Understanding currying

Creating control abstraction

Understanding *by-name* parameters

Project demo

