Reactive Scala #4

Cake pattern, path-dependent types, HList

Self type

Можно объявлять зависимости трейта на другие трейты. Это позволяет реализовать DI на уровне языка.

```
trait A

trait B {

   this: A =>

}
```

Difference with extends

Почему не подходит extends?

В этом случае если в A все методы реализованы, мы можем просто забыть добавить нужный mixin.

```
trait A {
  def foo = "It's foo 1"
}
trait B extends A
trait C extends A {
  override def foo = "It's foo 2"
}

new B //no compiler error
```

Difference with extends

То есть B не ведёт себя как отдельная компонента, что противоречит идеологии Dependency Injection.

Cake pattern

```
trait AComp {
  val a: A
  class A {
    def a() = "It's A"
  }
}
trait BComp {
  val b: B
  class B {
    def b() = "It's B"
  }
}
```

```
trait CComp {
  this: AComp with BComp =>
  def c() =
    "It's C with " + b.b() +
      " and " + c.c()
}
new CComp {
  val a = new A
  val b = new B
}
```

Path-dependent types

Напишем математически понятную имплементацию графа:

```
case class Node(id: Int)

case class Edge(left: Node, right: Node)

class Graph(nodes: ArrayBuffer[Node],
            edges: ArrayBuffer[Edge]) {
  def connect(left: Node, right: Node) {
    edges += Edge(left, right)
  }
}
```

Path-dependent types

Но лучше было бы запретить на уровне компилятора невозможные действия:

```
class Graph {
  case class Node(id: Int)
  case class Edge(left: Node, right: Node)
  val nodes: ArrayBuffer[Node] = ???
  val edges: ArrayBuffer[Edge] = ???
  def connect(left: Node, right: Node) {
    edges += Edge(left, right)
  }
}
```

Abstract types

Другой пример:

```
abstract class Key(id: String) {
  type Value
}
class DataStorage {
  val data = collection.mutable.Map[Key, Any]
  def get(key: Key): Option[key.Value] = ???
  def set(key: Key)(value: key.Value) = ???
}
object Keys {
  trait StringKey {
    this: Key =>
    type Value = String
  }
  val nameKey = new Key("name") with StringKey
}
```

Heterogeneous list

А теперь напишем HList, который состоит из HNil, HCons, а также есть метод concat, который умеет объединять два списка.