



EITHER

EITHER

```
Either[A, B]
```

"одно из двух", "исключающее или", "либо"

Имеет два подтипа

```
Left[A, B]
```

Левый вариант,
содержащий
значение типа A

```
Right[A, B]
```

Правый вариант,
содержащий
значение типа B

EITHER

```
val numOrStr1: Either[Double, String] = Left(2.12)
val numOrStr2: Either[Double, String] = Right("Scala")
```

EITHER

```
val numOrStr1: Either[Double, String] = Left(2.12)

val numOrStr2: Either[Double, String] = Right("Scala")

def info(numOrStr: Either[Double, String]): String =
  numOrStr match {
    case Left(num) => s"number $num"
    case Right(str) => s"string $str"
  }

info(numOrStr1) // number 2.12
info(numOrStr2) // string Scala
```

ОБРАБОТКА ОШИБОК

```
def sqrt(x: Double): Either[String, Double] =  
  if (x < 0) Left("negative number")  
  else Right(Math.sqrt(x))
```

ОБРАБОТКА ОШИБОК

Замена на значение по умолчанию: `getOrElse`

```
sqrt(7).getOrElse(0)
```

Фильтрация предикатом: `filterOrElse`

```
sqrt(3).filterOrElse(_ > 2, "too small")
```

ПРЕОБРАЗОВАНИЕ

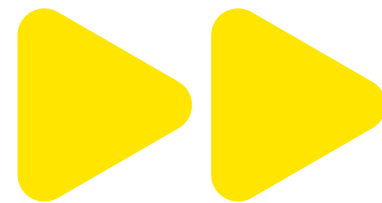
обычной функцией: `map`

```
sqrt(7).map(_.toString)
```

функцией, возвращающей `Either` с таким же "левым" типом:
`flatMap`

```
sqrt(7).flatMap(x => sqrt(x))
```

**В этом разделе
мы изучили Either**



**В следующем рассмотрим
коллекции**